

CLAIMS

We claim:

- 1 1). A method, comprising:
  - 2 optimizing an implementation of a programming language, comprising:
    - 3 analyzing one or more values computed by a program written in the
    - 4 programming language, wherein analyzing one or more values
    - 5 comprises;
    - 6 representing each bit within a value of the one or more values as an
    - 7 abstract element of a lattice having a set of abstract elements
    - 8 including  $0_A$ ,  $1_A$ ,  $\perp_A$  and  $T_A$ , wherein the lattice is an abstraction of a
    - 9 concrete domain containing 0, 1, and  $\perp$ ;
    - 10 analyzing one or more output bits that are produced by an operation in
    - 11 terms of one or more input bits that are input to the operation; and
    - 12 analyzing the input bits that are input to the operation in terms of the
    - 13 output bits that are produced by the operation.
  - 1 2). The method of claim 1, wherein optimizing further comprises:
    - 2 applying a forward abstract semantic to the abstract element; and
    - 3 applying a backward abstract semantic to the abstract element;
    - 4 wherein the forward abstract semantic is an approximation of a forward
    - 5 concrete semantic including AND, OR, and NOT; and
    - 6 wherein the backward abstract semantic is an approximation of a backward
    - 7 concrete semantic including  $AND^{-1}$ ,  $OR^{-1}$ , and  $NOT^{-1}$ .
  - 1 3). The method of claim 2, further comprising:
    - 2 identifying the values within the program as partially constant values.
  - 1 4). The method of claim 3, wherein the backward abstract semantic is for a
  - 2 complex boolean function including  $LEFT^{-1}$ ,  $URIGHT^{-1}$ ,  $JOIN^{-1}$ ,  $MEET^{-1}$ ,  $LE^{-1}$

3 and SRIGHT<sup>1</sup>, and wherein the forward abstract semantic is for the complex  
4 boolean function including LEFT, URIGHT, JOIN, MEET, LE, and SRIGHT.

1 5). The method of claim 4, wherein the program is represented in an  
2 intermediate language.

1 6). The method of claim 5, wherein the implementation is a compiler for the  
2 programming language.

1 7). The method of claim 5, wherein the implementation is a computer aided  
2 design compiler for the programming language.

1 8). A computer-readable medium having stored thereon a plurality of  
2 instructions, said plurality of instructions when executed by a computer,  
3 cause said computer to perform:  
4 optimizing an implementation of a programming language, comprising;  
5 analyzing one or more values computed by a program written in the  
6 programming language, wherein analyzing one or more values  
7 comprises;  
8 representing each bit within a value of the one or more values as an  
9 abstract element of a lattice having a set of abstract elements  
10 including  $0_A$ ,  $1_A$ ,  $\perp_A$  and  $T_A$ , wherein the lattice is an abstraction of a  
11 concrete domain containing 0, 1, and  $\perp$ ;  
12 analyzing one or more output bits that are produced by an operation in  
13 terms of one or more input bits that are input to the operation; and  
14 analyzing the input bits that are input to the operation in terms of the  
15 output bits that are produced by the operation.

1       9). The computer-readable medium of claim 8 having stored thereon additional  
2       instructions, said additional instructions when executed by a computer for  
3       optimizing, cause said computer to further perform:

4             applying a forward abstract semantic to the abstract element; and  
5             applying a backward abstract semantic to the abstract element;  
6             wherein the forward abstract semantic is an approximation of a forward  
7                 concrete semantic including AND, OR, and NOT; and  
8             wherein the backward abstract semantic is an approximation of a backward  
9                 concrete semantic including AND<sup>-1</sup>, OR<sup>-1</sup>, and NOT<sup>-1</sup>.

1       10). The computer-readable medium of claim 9 having stored thereon  
2       additional instructions, said additional instructions when executed by a computer,  
3       cause said computer to further perform:  
4             identifying the values within the program as partially constant values.

1       11). The computer-readable medium of claim 10, wherein the backward  
2       abstract semantic is for a complex boolean function including LEFT<sup>-1</sup>,  
3       URIGHT<sup>-1</sup>, JOIN<sup>-1</sup>, MEET<sup>-1</sup>, LE<sup>-1</sup> and SRIGHT<sup>-1</sup>, and wherein the forward  
4       abstract semantic is for the complex boolean function including LEFT,  
5       URIGHT, JOIN, MEET, LE, and SRIGHT.

1       12). The computer-readable medium of claim 11, wherein the program is  
2       represented in an intermediate language.

1       13). The computer-readable medium of claim 11, wherein the implementation  
2       is a computer aided design compiler for the programming language.

1       14). A system, comprising:  
2             a processor;

3 memory connected to the processor storing instructions for bidirectional  
4 bitwise constant propagation by abstract interpretation executed by the  
5 processor;  
6 storage connected to the processor that stores a software program having a  
7 plurality of separately compilable routines,  
8 wherein the processor optimizes an implementation of a programming  
9 language, by  
10 analyzing one or more values computed by a program written in the  
11 programming language, wherein analyzing one or more values  
12 comprises;  
13 representing each bit within a value of the one or more values as an  
14 abstract element of a lattice having a set of abstract elements  
15 including  $0_A$ ,  $1_A$ ,  $\perp_A$  and  $T_A$ , wherein the lattice is an abstraction of a  
16 concrete domain containing 0, 1, and  $\perp$ ;  
17 analyzing one or more output bits that are produced by an operation in  
18 terms of one or more input bits that are input to the operation; and  
19 analyzing the input bits that are input to the operation in terms of the  
20 output bits that are produced by the operation.

1 15). The system of claim 14, wherein the processor further optimizes by  
2 applying a forward abstract semantic to the abstract element; and  
3 applying a backward abstract semantic to the abstract element;  
4 wherein the forward abstract semantic is an approximation of a forward  
5 concrete semantic including AND, OR, and NOT; and  
6 wherein the backward abstract semantic is an approximation of a backward  
7 concrete semantic including  $AND^1$ ,  $OR^1$ , and  $NOT^1$ .

1 16). The system of claim 15, wherein the processor identifies the values within  
2 the program as partially constant values.

1    17). The system of claim 16, wherein the backward abstract semantic is for a  
2    complex boolean function including LEFT<sup>1</sup>, URIGHT<sup>1</sup>, JOIN<sup>1</sup>, MEET<sup>1</sup>, LE<sup>-1</sup>  
3    and SRIGHT<sup>1</sup>, and wherein the forward abstract semantic is for the complex  
4    boolean function including LEFT, URIGHT, JOIN, MEET, LE, and SRIGHT.

1    18). The system of claim 17, wherein the program is represented in an  
2    intermediate language.

1    19). The system of claim 18, wherein the implementation is a compiler for the  
2    programming language.

1    20). The system of claim 19, wherein the implementation is a computer aided  
2    design compiler for the programming language.

1    21). A system, comprising:  
2       means for optimizing an implementation of a programming language,  
3           comprising;  
4       means for analyzing one or more values computed by a program written  
5           in the programming language, wherein analyzing one or more values  
6           comprises;  
7       means for representing each bit within a value of the one or more  
8           values as an abstract element of a lattice having a set of abstract  
9           elements including 0<sub>A</sub>, 1<sub>A</sub>, ⊥<sub>A</sub> and T<sub>A</sub>, wherein the lattice is an  
10          abstraction of a concrete domain containing 0, 1, and ⊥;  
11       means for analyzing one or more output bits that are produced by an  
12          operation in terms of one or more input bits that are input to the  
13          operation; and

14 means for analyzing the input bits that are input to the operation in  
15 terms of the output bits that are produced by the operation.

1 22). The system of claim 21, wherein the means for optimizing further  
2 comprises:

3 means for applying a forward abstract semantic to the abstract element; and  
4 means for applying a backward abstract semantic to the abstract element;  
5 wherein the forward abstract semantic is an approximation of a forward  
6 concrete semantic including AND, OR, and NOT; and  
7 wherein the backward abstract semantic is an approximation of a backward  
8 concrete semantic including AND<sup>-1</sup>, OR<sup>-1</sup>, and NOT<sup>-1</sup>.

1 23). The system of claim 22, further comprising:

2 means for identifying the values within the program as partially constant  
3 values.

1 24). The system of claim 23, wherein the backward abstract semantic is for a  
2 complex boolean function including LEFT<sup>-1</sup>, URIGHT<sup>-1</sup>, JOIN<sup>-1</sup>, MEET<sup>-1</sup>, LE<sup>-1</sup>  
3 and SRIGHT<sup>-1</sup>, and wherein the forward abstract semantic is for the complex  
4 boolean function including LEFT, URIGHT, JOIN, MEET, LE, and SRIGHT.

1 25). The system of claim 24, wherein the program is represented in an  
2 intermediate language.

1 26). The system of claim 25, wherein the implementation is a compiler for the  
2 programming language.

1 27). The system of claim 26, wherein the implementation is a computer aided  
2 design compiler for the programming language.